

js的学习:

js的概念和声明

js的变量

js的运算符和逻辑结构

js的数组

js的函数(常用的函数和对象)

js的事件机制

js的window对象

js的document对象

前情回顾

每日鸡汤:

越努力,越幸运!

CSS学习:

css的概念

css的声明

css的选择器

css的常用属性

css的定位

css的盒子模型

作用:

在HTML网页中添加样式,让网页看起来更加的优美.

js 的概念和声明

Js的概念和声明:

问题:

在网页的发展历程中,发现网页不能对用户的数据进行自动校验,和

提供一些特效

造成用户体验极差

解决:

使用JavaScript

作用:

可以让网页和用户之间进行直接简单的交互.

可以给网页制作特效和动画

注意:

js是由浏览器解析执行的.

js需要在HTML文档中进行声明

使用:

声明Js代码域

1.在head标签中使用script声明js代码域

```
<head>

....

<!--声明js代码域-->
<script type="text/javascript">
    function test(){
        alert("哈哈,js学习起来很简单!!!");
    }
</script>
</head>
```

2.在head标签中使用script引入外部声明的js文件

```
<head>

....

<!--声明js代码域-->

<!--引入外部声明好的js文件-->
    <script src="js/my.js"
type="text/javascript" charset="utf-8"></script>
</head>
```

js 的变量

js的变量学习:

```
<html>
<head>

    <title>js的变量学习</title>
```

```
<meta charset="UTF-8"/>  
<!--
```

js的变量学习:

1 js的变量声明

使用var关键字进行变量声明,格式如下:

```
var 变量名=初始值;
```

例如: var a="呵呵";

2 js变量的特点

a、 变量声明只有var关键字,声明的变量可以存储任意类型的数据。

b、 js中的代码可以不适用分号结尾,但是为了提升代码的阅读性,建议使用分号。

c、 js中的变量允许出现同名变量,但是后面的会将前面的覆盖。

d、 声明不赋值,默认值是undefined

3 js的数据类型

使用换件typeof判断变量的数据类型

number:数值类型

string:字符类型,注意:在js中字符可以使用单引号也可以使用双引号

boolean:布尔类型

object:对象类型

4 js的变量强转

使用Number()函数：将其他数据类型转换为数值类型，转换失败返回NaN(not a number)

使用Boolean()函数：将其他数据类型转换为布尔类型，有值返回true，无值返回false；

5 特殊的值

null	object
undefined	undefined
NaN	number

作用：

变量是用来存储数据的,方便程序进行操作.

-->

<!--声明js代码域-->

<script type="text/javascript">

//js的变量声明

```
var a=1;
var a1=2.2;

var a3='哈哈';

var a31='和';
var a4=false;
var a5=new Date();
//alert(a3);

var b="北京八分钟";

var b="平昌冬奥会";

//alert(b);
var c;
//alert(a3);
```

//js的数据类型

```
//alert(typeof a5);
```

//js的变量强转

```
        var a=1;
        var b="11";

        var c="哈哈";

        var d;
        alert(typeof null);
        alert(Boolean(d));
        if(Boolean(a)){
            alert("js学习");
        }
    </script>
</head>
<body>

    <h3>js的变量学习</h3>

    <hr />
</body>
</html>
```

js 的运算符和逻辑结构

js的运算符和逻辑结构：

作用：变量是存储要处理的数据的，运算符和逻辑结构就是用来处理数据的

```
<html>
  <head>

    <title>js的运算符和逻辑结构</title>

    <meta charset="UTF-8"/>
    <!--
```

js的运算符和逻辑结构：

作用：结合变量进行数据处理

使用：

算术运算符：

`+, -, *, /, %`

关系运算符：

`>, >=, <, <=, !=`

等值符：`==`

如果类型一致则直接比较值

如果类型不一致则先使用Number强转为同一类型后

再比较值

等同符：`===`

先判断类型，如果类型一致则再比较内容

如果类型不一致则直接false

逻辑运算符：

`&& || !`

逻辑结构：

```
if(){}else if(){} else{}  
switch(){}  
for(){}  
while(){}  
do{}while()
```

注意：

判断条件可以直接是变量。

注意：

js中变量是没有类型的，但是数据是有类型的，在进行数据

处理的时候

要注意数据的类型。

```
-->
<!--声明js代码域-->
<script type="text/javascript">

    //算术运算符

    var a=3;
    var b=4;
    //alert(a+b);

    //关系运算符

    var a1=6;
    var b1=5;
    //alert(a1>b1);

    //逻辑结构

    //alert(a1>b1&&a>b);
/*-----*/

    //等值符(==)

    var a2=1;
    var a3=1;
    var b2="1";
    var c2=true;
    var d2="true";
    var e2="a";
//    alert(a2===a3);//true        true
//    alert(a2===b2);//true        false
//    alert(a2===c2);//true        false
//    alert(a2===d2);//false       false
//    alert(b2===c2);//true        false
//    alert(b2===d2);//false       fasle
//    alert(d2===e2)//false       false
/*-----*/

    //for循环结构

    for(var i=0;i<5;i++){

        //alert("该吃饭了,好饿");

    }

    //九九乘法表

    for(var i=1;i<=9;i++){
        for(var j=1;j<=i;j++){
```



```
document.write(j+"*"+i+"="+i*j+"&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;");
    }
    document.write("<br />");
}

</script>
</head>
<body>

    <h3>js的运算符和逻辑结构</h3>

    <hr />
</body>
</html>
```

js 的数组

js的数组学习：

作用：存储数据，保证数据的完整性，操作方便。

```
<html>
  <head>

    <title>js的数组</title>

    <meta charset="UTF-8"/>
    <!--
```

js的数组学习：

问题：

使用变量存储数据，如果数据量比较大的时候，会造成需要

声明

大量的变量去存储数据，代码整洁度及阅读性极差，数据的

完整性得不到保证。

解决：

使用数组

作用：

存储数据

使用：

js的数组的声明

```
var arr1=new Array();//声明一个空数组
```

```
var arr2=new Array(长度)//声明指定长度的数组。
```

```
var arr3=[]//声明一个空数组，也可以在声明时直接
```

赋值，例如：`var arr3=["a","b","c"];`

js数组的使用

数组的赋值

数组名[角标]=值；

注意：js中赋值可以跳跃角标赋值，不存在的角标

也可以赋值，会对数组的大小进行改变。

数组的取值：

```
var 变量名=数组名[角标名]
```

注意：如果获取的角标没有数据，则返回

`undefined;`

js的数组的特点

特点1：

js中的数组可以存储任意类型的数据。

特点2：

js的数组可以通过length属性动态的改变长度。

可以增加，也可以缩短。

注意：

如果是增加，则使用逗号进行占位

如果是缩减则从后往前减少存储的数据。

js的数组的遍历：

普通for循环

增强for循环

注意：增强for循环中，循环条件声明的变量记录的是角标。

js的数组的常用操作方法：

数组名.pop()//移除并返回最后一个元素。

数组名.push(要添加的数据)//在数组最后追加数据，并返回新的长度。

```
-->
```

```
<!--声明js代码域-->
```

```
<script type="text/javascript">
```

```
//创建数组
```

```
//第一种方式
```

```
var arr1=new Array();
```

```
arr1[0]="哈哈";
```

```
arr1[1]="嘿嘿";
```

```
//alert(typeof arr1);
```

```
//alert(arr1.length);

//第二种声明方式
var arr2=new Array(5);
var arr4=new Array([5,7,8]);
//alert(arr4);

//第三种声明方式
var arr3=["a","b","c"];
//alert(arr3);

//使用数组
var arr=[];

//给数组的赋值

arr[0]="张三";

arr[1]="李四";

arr[10]="王五";
//alert(arr)
//获取数组中的数据
var a=arr[0];
var b=arr[100];
//alert(b);

//数组的特点

//可以存储任意类型的数据
var arr5=[1,"a",new Date()];
//alert(arr5.length);
arr5.length=10;
//alert(arr5);
arr5.length=2;
//alert(arr5);

//数组的遍历

var arr=[1,2,3,4,5];

//第一种：普通for循环
//
// for(var i=0;i<arr.length;i++){
//     alert(arr[i]);
// }
```

```
//      }  
  
      //第二种：高级for  
//      for(var i in arr){  
//          alert(arr[i]);  
//      }  
  
      //常用的操作方法：  
  
      var arr=["a","b","c","d"];  
      var str=arr.pop();  
      alert(str);  
      alert(arr);  
  
      var str2=arr.push("哈哈");  
      alert(str2);  
      alert(arr);  
    </script>  
  </head>  
  <body>  
    <h3>js的数组学习</h3>  
    <hr />  
  </body>  
</html>
```

js 的函数学习

js的函数学习：

作用:封装功能代码，降低代码冗余。


```
<html>  
  <head>  
    <title>js的函数学习</title>  
    <meta charset="UTF-8"/>  
    <!--  
      js的函数学习：
```

问题：

其实开发就是对现实生活中的问题使用代码进行解决，同类型的问题非常多，这样就需要每次都把代码重新声明一遍，造成代码过于冗余。

解决：

封装成函数，不用重复声明，调用即可。

使用：

函数的声明：

第一种声明方式：

```
function 函数名(形参1,形参2,...){函数体....}
```

第二种声明方式：

```
var 变量名=new Function("形参名1","形参名2",..., "函数体");
```

注意：

在js中函数是作为对象存在的。

第三种声明方式：

```
var 变量名=function()(形参1,形参2,...){函数体....}
```

函数的形参：

在js中函数的形参在调用的时候可以不赋值，不报错，

但是默认为undefined

在js中函数的形参在调用的时候可以不完全赋值，依次赋值。

注意：

js中没有函数重载，只有函数覆盖。

函数的返回值：

在函数内部直接使用return语句返回即可，不需要返回值类型

注意：

默认返回undefined;

函数的调用：

1、在加上代码域中直接调用(主要进行页面资源初始化)

2、使用事件机制(主要实现和用户之间的互动)

3、作为实参传递(主要是动态的调用函数)

注意：

小括号为函数的执行符，函数名()才会被执行，直接函数名则作为对象使用。

注意：

js的代码区域只有一个,包括引入的js代码，浏览器会将引入的js文件和内部声明的js代码解析成一个文件执行。

js代码的调用和声明都在一个区域内。

```
-->

<!--声明js代码域-->

<script src="js/my.js" type="text/javascript"
charset="utf-8"></script>
<script type="text/javascript">

    //函数的声明

    //第一种声明方式
    function test1(a,b){
        alert("我是第一种声明方式"+(a+b));
    }

    //调用
    test1(1,2);

    //第二种声明方式
    var test2=new Function("a","b","alert(a+b);");
    test2("哈哈","嘿嘿");

    //第三种声明方式
    var test3=function(a,b){
        alert("我是第三种声明方式"+a+b);
    }

    var test3=function(a,b,c){
        alert("我是第三种声明方式"+a+b);
    }
    test3("6666","3333");

    //函数的形参

    //声明函数
    function demo(a,b){
        alert("函数的形参学习"+a+b);
    }
    demo();

    demo("哈哈");
```


//函数的返回值

//声明函数

```
function demo2(a,b){
```

```
}
```

```
var str=demo2("js","的返回值");
```

```
alert(str);
```

//函数的调用

//声明函数

```
var demo3=function(a,b){
```

```
    alert("函数的调用")
```

```
}
```

//1.js代码域中直接调用

```
demo3();
```

//2.使用事件机制(可以根据用户在页面的不同操作来触发不

同的函数执行)

//3.作为实参传递

```
function demo4(a){
```

```
    a();
```

```
    alert(a);
```

```
}
```

```
demo4(demo3);
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<h3>js的函数学习</h3>
```

```
<hr />
```

```
</body>
```

```
</html>
```

js 的事件机制

js的事件机制学习：

作用：根据用户的行为触发响应的js函数的执行，实现网页和用户之间的互动。

```
<html>
  <head>
    <title>js的事件机制学习</title>
    <meta charset="UTF-8"/>
    <!--
```

课堂小结：

声明：声明js代码域用来在html中书写js代码

变量：记录要运算的数据。只有var关键字，同名会覆盖，可以存储任意类型的数据

运算符和逻辑结构：用来处理变量记录的数据。和java使用方式基本一致，注意条件声明使用var

数组：存储大量需要运算的数据，保证数据的完整性

函数：封装功能。函数也是对象，在js中函数可以作为参数传递。

js的事件机制：

概念：基于监听的。一个动作会触发其他事物的执行。

作用：根据用户在网页中的不同操作触发执行对应的功能函数。

使用：

单双击事件：

单击事件：onclick.

双击事件：ondblclick

鼠标移动事件：

鼠标悬停事件：onmouseover 当鼠标在HTML元素之

上时触发

鼠标移出事件：onmouseout 当鼠标移出某个HTML

元素时触发

键盘事件：

键盘下压事件：onkeydown 当键盘被按下时触发

键盘弹起事件：onkeyup 当键盘弹起时触发

焦点事件：

获取焦点：onfocus 当获取焦点时触发

失去焦点：onblur 当失去焦点时触发

值改变事件：

专门给select标签使用：

onchange事件 当下拉框的值改变时触发

页面加载事件：

专门给body标签使用

onload 当页面加载成功后触发

注意：

事件是作为HTML标签的属性来使用的。

一个HTML元素可以同时使用多个事件，但是注意事件之间的相

互干扰。

```
-->
<script type="text/javascript">
//单双击事件

    //创建函数
    function testCilck(){
        alert("我是单击");
    }

    function testDbClick(){
        alert("我是双击");
    }
//鼠标移动事件
    function testMouseOver(){
        alert("我进来啦");
    }
    function testMouseOut(){
        alert("我出来啦");
    }
//键盘事件
    function testKeyDown(){
        alert("我被按下啦");
    }
    function testKeyUp(){
        alert("我起来啦");
    }
//焦点事件
    function testFocus(){
        alert("我获取焦点啦");
    }
    function testBlur(){
```

```
        alert("我失去焦点啦");
    }
    //值改变事件
    function testChange(){
        alert("我被改变啦");
    }
    //页面加载事件
    function testLoad(){
        alert("我加载成功啦");
    }
</script>
<!--声明css代码域-->
<style type="text/css">

    #div01{
        border: solid 1px;
        width: 200px;
        height: 200px;
        background-color: orange;
    }
</style>
</head>
<body onload="testLoad()">

    <h3>js的事件机制</h3>

    <hr />

    <input type="button" value="测试单击"
onclick="testCilck()"/>

    <input type="button" value="测试双击"
ondblclick="testDbClick()" />
    <hr />

    <h4>鼠标事件:</h4>

    <div id="div01" onmouseover="testMouseOver()"
onmouseout="testMouseOut()">
```

[illegible]

js 计算器小练习

js计算器小练习:

```
<html>
  <head>
    <title>js计算器</title>
```

```
<meta charset="UTF-8"/>
<!--声明css代码域-->
<style type="text/css">
/*设置div样式*/
    #math{
        text-align: center;
        margin: auto;
        margin-top: 100px;
        border: solid 1px;
        border-radius: 10px;
        width: 300px;
        height: 400px;

    }
/*设置文本框*/
    #num{
        width: 270px;
        height: 40px;
        margin-top: 20px;
    }
/*设置按钮样式*/
    input[type=button]{
        border: solid 1px orange;
        border-radius: 10px;
        width: 60px;
        height: 60px;
        margin: 5px;
        margin-top: 16px;
        font-size: 30px;
    }
</style>
<!--声明js代码域-->
<script type="text/javascript">
    //声明函数
    function operNum(num){
        //获取文本框
        var inp=document.getElementById("num");
```

```
//使用switch进行判断
switch (num){
    case "=":
        inp.value=eval(inp.value);
        break;
    case "C":
        inp.value="";
        break;
    default:
        inp.value=inp.value+num;
        break;
}
}
</script>
</head>
<body>
    <div id="math">
        <input type="text" name="num" id="num" value="" /><br />
        <input type="button" id="" value="1"
onclick="operNum(this.value)"/>
        <input type="button" id="" value="2"
onclick="operNum(this.value)"/>
        <input type="button" id="" value="3"
onclick="operNum(this.value)"/>
        <input type="button" id="" value="4"
onclick="operNum(this.value)"/><br />
        <input type="button" id="" value="5"
onclick="operNum(this.value)"/>
        <input type="button" id="" value="6"
onclick="operNum(this.value)"/>
        <input type="button" id="" value="7"
onclick="operNum(this.value)"/>
        <input type="button" id="" value="8"
onclick="operNum(this.value)"/><br />
        <input type="button" id="" value="9"
onclick="operNum(this.value)"/>
        <input type="button" id="" value="0"
onclick="operNum(this.value)"/>
        <input type="button" id="" value="+"
onclick="operNum(this.value)"/>
        <input type="button" id="" value="-"
onclick="operNum(this.value)"/><br />
        <input type="button" id="" value="*"

```



```
onclick="operNum(this.value)"/>
    <input type="button" id="" value="/"
onclick="operNum(this.value)"/>
    <input type="button" id="" value="C"
onclick="operNum(this.value)"/>
    <input type="button" id="" value="="
onclick="operNum(this.value)"/>
    </div>
</body>
</html>
```